UNIVERSITY OF WINDSOR

ELECTRICAL ENGINEERING CO-OP

WEB SECURITY BEST PRACTICES

BLACKBERRY

SOFTWARE DEVELOPMENT

WATERLOO, ON

Submitted to: Mr. Byron Guptill

Submitted by: Eric Parker

Submitted on: May 15, 2017

Work Term Number: Winter 2017

University of Windsor
Faculty of Electrical and Computer Engineering
Windsor, ON N9B 3P4

May 15, 2017

Mr. Byron Guptill
Manager, Portals & HR
2200 University Ave. E
Waterloo, ON N2K 0A2

Dear Mr. Guptill,

Please accept this report entitled “Web Security Best Practices” as my submission to fulfil my work term requirements.

It was an honour to complete my second work term at BlackBerry with the Portals & HR team. It was my pleasure to have the opportunity to participate in the company’s pivot and work on a wide variety of projects that were important to the success of the corporation. During my time at BlackBerry, I was fortunate enough to further develop a strong technical background while also understanding the business impact of my actions. I gained valuable work experience through contributing to a number of different projects and learning directly from you during our one-on-one meetings. The projects I worked on awarded me the opportunity to work with several very intelligent and highly skilled people from which I learned a great deal.

Overall, this work term has provided me with valuable work experience, a deeper passion for computer programming, and a better understanding of what drives decision-making in the tech industry. My submitted report outlines the importance of producing and maintaining secure web applications, and measures that must be taken when handling sensitive customer data over the web.

Finally, I would like to thank you, my manager, for sharing your experience and knowledge with me throughout my work term. I enjoyed learning from you and I am very grateful that you were willing to share your time with me and answer my questions, not to mention I had a great time with the team!

Sincerely,

Eric Parker

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

Executive Summary:

The purpose of this report is to exemplify several common ways in which hackers can potentially exploit software applications to attain sensitive information or alter business data, and countermeasures that businesses can implement in order to prevent such attacks. The ever-increasing role that hacking plays in society is powerful; Hillary Clinton even attributes her election loss to Donald Trump in part due to the work of Russian hackers. With this increased hacker influence, software developers and management teams need to ensure that all code written is done so securely, placing extra emphasis on ensuring proper security measures are taken for even the smallest software applications. Security risks are heightened when information is passed through the internet, and regardless of which business sector a business is in, web presence is particularly important to the success of a business in the mobile economy that exists today. This means vast amounts of data are collected each year by almost every single company in the world via web services. Such information often includes highly sensitive information including credit card numbers, passwords, and personal information.

For this reason, it is particularly important that web developers understand hacking techniques in order to prevent them. This report outlines some of the most effective and common hacking techniques used today and how web developers and software developers alike can make simple changes to their code in order prevent such attacks.

Introduction:

This report outlines the web application security techniques used by BlackBerry, a global leader in web and mobile security, to ensure customer information is passed over the internet and stored securely. In 2015, there were 177,866,236 personal records exposed that were held by financial institutions, educational institutions, health or medical institutions, businesses, the military, or the government resulting in over $445 billion in losses. According to WhiteHat Security, web application attacks represent the greatest threat to an organization's security.

BlackBerry mobile security solutions are deployed, tested, and trusted in many of the world's largest and most demanding mobile environments requiring maximum security. BlackBerry security solutions have been deployed and tested extensively in some of the most demanding mobile environments in the world, including usage in over half of the Fortune 100 businesses and 100% of the Fortune 100 Aerospace and Defense firms. Naturally, web security is taken very seriously at BlackBerry, and it is imperative that ever software developer is well informed in the field of hacker prevention.

According to WhiteHat Security, a leader in the Gartner Magic Quadrant for Application Security Testing, web application vulnerabilities can be divided into different "classes" or categories of attacks, each of which having their own unique attributes. For example, Cross-Site Scripting (XSS) attacks are a type of injection in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Each year, WhiteHat Security publishes the "Web Applications Security Statistics

Report", which uses data collected from tens of thousands of websites to reveal common security vulnerabilities associated web applications deployed throughout all industries. The report's findings are based on the aggregated vulnerability scanning and remediation data from web applications that use the *WhiteHat Sentinel* service for security testing, which is used by 20,000 businesses in a variety of industries. Figure 1, taken from this report, shows the top 5 most common website security vulnerabilities by class.

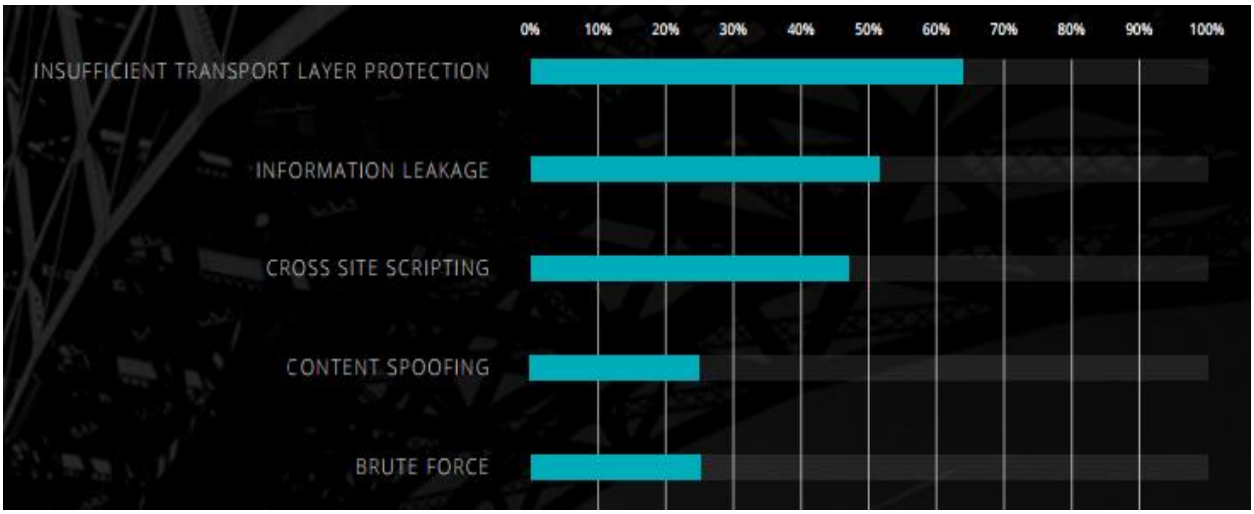

Figure 1: WhiteHat Security – Vulnerability Likelihood by Class

As seen in Figure 1, the top 5 website vulnerabilities are insufficient transport layer protection (ITLP), information leakage, cross-site scripting (XSS), content spoofing, and brute force hacking. As can be seen in Figure 1, the percentages of websites that contain such vulnerabilities are shockingly high, with the top two vulnerabilities present in over half of all websites tested! This report will focus on outlining and explaining each of these issues in detail, followed by steps that can be taken by development teams in order to prevent the possibility/probability of each attack. Please note that content spoofing will (which is a type of exploit used by a malicious hackers to present a faked or modified web site to the user as if it were legitimate – for

example if a hacker created a website that looked exactly like PayPal and sent out emails to PayPal users prompting them to login to “PayPal” but providing them with the link to their fake website to try to get people to enter their real credentials) will not be covered in this report because it is a malicious practice that is not entirely preventable via software. Prevention techniques include having customer service representatives available via phone and/or email so that customers can contact the business in the event that a customer believes they have been the victim of content spoofing; this is outside the scope of this report. In its place, SQL injection will be covered instead as this is a very common, dangerous, and easily preventable vulnerability that every software developer should be aware of.

Overview

BlackBerry, formerly known as Research In Motion (RIM), is a Canadian-based multinational software and wireless telecommunications company founded in Waterloo, Ontario. BlackBerry operates in 77 cities through 32 countries around the world with headquarters located in Canada, United States, United Kingdom, Germany, and Singapore. They employ over $4{,}500$ employees including over 200 co-op students, and offer products in a number of different tech sectors including mobile phones and devices, real-time operating systems, internet of things (IOT), automotive, and enterprise software. BlackBerry is an industry leader in security with many worldwide certifications to back it up.

One of the primary technical divisions of BlackBerry contributing to the enterprise security software suite offered by BlackBerry is Business Application Services (BAS). This division is responsible for managing and delivering a number of web-based customer relationship management (CRM) applications that handle everything from sales transactions to online customer service. Since great emphasis is placed on mobile and web security at BlackBerry, it is the responsibility of the BAS team to withhold such a high security standard with every software application they develop. For this reason, this report will cover in detail web security best practices used at BlackBerry to prevent against some of the most common and dangerous hacker attacks.

This report will cover five of the most commonly occurring security vulnerabilities present on the web today as well as the security measures that must be in place in order to ensure site and user information is kept safe.

In order to understand the contents of this report, it is necessary to have a basic understanding of how information is communicated over the internet. The internet is a global computer network providing a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols. The standard communication protocol used by the world wide web is Hypertext Transfer Protocol (HTTP). Communication protocols (such as HTTP) are formal descriptions of digital message formats and rules, and they are very important in telecommunications (in fact they are required) because both sides must be "speaking the same language" in order to communicate. For example, when writing a letter to someone in Canada, in order to make sure that letter gets to the correct person you must include the receivers name, address, province, and postal code on the envelope so that the post office can interpret who the letter is for. This information is laid out in a particular manner so that the post office workers understand how to interpret such information. This is an example of (a very informal) communication protocol. The HTTP protocol defines similar information as well as covers authentication, error detection and correction, and signaling, and must be adhered to in order to make a request over the internet. All communications are done between a web browser (the client) and a website (the server).

What does this mean for web security? As with any telecommunication channel, transferring information in this manner allows for unwanted people to detect and potentially manipulate such messages. Thus, it is the job of the website developers to ensure that the appropriate security measures are in place to ensure that HTTP communications can be transacted safely and securely so that user information is kept private.

I. Insufficient Transport Layer Protection

Insufficient Transport Layer Protection (ITLP) is the most common website vulnerability. According to WhiteHat Security, such vulnerabilities are present in over 63% of all websites. ITLP vulnerabilities are so common because there are many ways in which they can come about, but all of which are related to the absence of transport encryption at some point within the website when sensitive information is being passed over the internet. As discussed previously, information is passed over the internet using the HTTP protocol. This communication protocol defines an agreed-upon format in which information should be transported, but does nothing to protect the information itself. That is, anyone on the internet is capable of monitoring network traffic, meaning they are capable of getting all information transmitted in the HTTP requests performed by all users on that network because the information is written as text that anyone can read. This is a huge problem if sensitive information (including password or credit card information) is being transmitted over the network because a hacker can easily view intercept information.

The solution to this problem is very straightforward: use Hypertext Transfer Protocol Secure (HTTPS) for all pages requiring sensitive information. HTTPS is a version of HTTP that ensures all communications on the web between the client and the server are encrypted (information is converted into a secret code that can only be understood by authorized users – it is no longer plain text that can be read by anyone). HTTPS uses a security protocol called Secure Sockets Layer (SSL) which uses asymmetric public key infrastructure. The details of how this encryption works will not be discussed since it is outside the scope of this report. The main takeaway is

websites need encryption to ensure hackers are not stealing sensitive information simply by listening to the network.

In order for a website to use the HTTPS protocol, a few things need to be done by the development team. First, the company that owns the website must purchase an SSL certificate which acts as a unique identifier for a website and contains information unique to that business that will be used in the encryption process. Next, the SSL certificate must be activated and installed on the web server, and the server needs to be configured to use the HTTPS protocol instead of HTTP. That's it! That is all that needs to be done in order to prevent against ITLP vulnerabilities. Since this vulnerability is so easy to fix, why is this problem so common?

ITLP vulnerabilities can be introduced in a number of different ways. Besides the obvious unsafe circumstance in which a website does not use the HTTPS protocol for all pages that require authentication, there are other ways ITLP vulnerabilities can present themselves. Take for example the (common) circumstance in which a website has an improperly configured SSL certificate that causes browser warnings for its users. In most circumstances, the user will dismiss the browser warnings and continue using the website. The user becomes accustomed to dismissing such warnings, never taking the time to learn about what the warnings mean. This type of user behaviour makes them especially susceptible to phishing attacks since they are used to dismissing security warnings about improperly configured SSL certificates. As another example, websites may be using HTTPS for serving files, but fail to use encrypted connections to databases, rendering all transactions between the web service and its databases vulnerable. The main takeaway for developers and Information Technology (IT) operations: Ensure HTTPS is

used for all pages with sensitive information, and ensure that the SSL certificate in use is properly configured and not expired.

II. Information Leakage

The second most common website vulnerability is known as information leakage, occurring in over 51% of all websites. Information leakage occurs when an application directly reveals any form of sensitive information including details about the web application or web server, environment, or even user-specific data. Such errors most commonly occur due to bad software design or failure to review code prior to public deployment; two very common circumstances that must avoided by developers.

One of the most common (and dangerous) forms of information leakage occurs when developers leave comments in their code for personal or internal reference, but these comments are not removed before the code is deployed to public servers, thus allowing hackers to review the public source code and read these comments in order to attain information about internal systems or servers. For example, a developer might insert the following comment during development: “Make sure to restart server 192.168.0.110 every time changes are made to the configuration file”. Although the comment provides the developer with a helpful reminder to restart the server after making configuration changes (which might save them time when debugging during development), if the developer forgets to delete this comment before deploying the code to production, they have just leaked the internal IP address of the web server to everyone on the internet. This is valuable information for a hacker to have. Since developers are often working on large scale applications with thousands of lines of code, comments like this one are often overlooked making vulnerabilities of this sort far too common. In order to avoid information leakage in this manner, developers should be careful

about the type of information they are putting in comments, and instead include all comments that would have contained sensitive information in a separate file that will not be deployed.

Another very common form of information leakage occurs when websites are far too descriptive with error messages. During development, it is often useful to have web services return very descriptive error messages, often containing very sensitive information about database queries or connection information, when errors occur so that problems are easy to debug and fix. However, if these detailed error messages are not substituted with very generic error messages before deployment, hackers have immediate access to extremely sensitive information and can often exploit the messages to help them perform malicious attacks. In order to avoid information leakage of this type, developers need to be very conscious of the repercussions of sharing such information. As a general rule, the lazier a developer is during development, the more susceptible a web application is to information leakage. Developers must ensure that all error messages presented by production applications are very vague and general so that the messages can not be used by hackers to attain system information.

III. Cross-Site Scripting

Cross-site scripting (XSS) is a very common security vulnerability exploited by hackers to steal valuable customer information from an otherwise secure website. Even if a website has ensured proper measures to protect against ITLP and information leakage vulnerabilities, customer data could still be stolen via XSS. XSS is a particularly malicious type of injection in which hacker code is injected directly into otherwise benign and trusted websites. XSS attacks occur when a hacker uses a web application to send malicious code (usually a browser side script) to a different user. Hackers are able to inject their own code into another website typically by using input fields to inject code into a website's database. For example, consider a blog-type website that does not protect against XSS and features a comment section at the bottom where anonymous users can leave comments about the content of the blog. Instead of typing a legitimate comment as they should, the hacker types the following into the comment section and posts the "comment": *<script type="text/javascript">alert(document.cookie);</script>*. The "comment" that the hacker just introduced into the website contains JavaScript code that will alert the webpage's cookies (which often contain very sensitive customer information) for the user to see. This is because the hacker has injected HTML/JavaScript code into the comment section of the website, so when it is rendered by the web browser, the "comment" will be treated as JavaScript code instead of text to be displayed as a comment. This is extremely dangerous because the hacker can exploit this to add literally any code they want to your website, and this code will show for all other users of the website, not just the hacker. For this reason, XSS is particularly dangerous and must be prevented.

There are a number of ways that developers can prevent against XSS, however many of these solutions are specific to the programming languages and technologies being used for the web application itself, so prevention techniques will be discussed in a general manner. The first way to prevent against XSS attacks is to perform input validation on all user inputs which limits the set of allowable user inputs. This can be done by establishing a set of restricted characters called a *blacklist*, which contains all characters or sequences of characters that an input cannot contain. Considering the previous example, the website developer would want to blacklist the sequence "<script>" and "</script>" to ensure that no person who writes a comment is able to inject JavaScript into the page. It should be noted that blacklist validation only works if the developer includes *every* potentially dangerous character or sequence of characters in the blacklist which is not possible. Thus, blacklists should be combined with other techniques, such as *encoding*, to add additional security. Encoding involves making direct transformations to a user's input so that it cannot be interpreted as code. This is accomplished through the use of Hypertext Markup Language (HTML) character entities which are strings of non-special characters that correspond to each special symbol. This helps the browser determine the difference between actual code on the website and the user's input, all while being visually rendered the same way by the browser. There are many different ways to accomplish this, most of which are programming language specific. The main takeaway for developers: ensure that all user inputs are validated and all special characters are escaped and/or converted to HTML character entities before the information is saved in a database.

IV. Brute Force

The brute force vulnerability refers to a trial-and-error method that allows hackers to try all possible combinations of information in a user input (such as a username or a personal identification number (PIN)) until the correct combination of characters is achieved. Since computers are capable of executing code extremely fast, is it very easy for a hacker to write a software script that tries all possible combinations of characters on a keyboard until a correct password is found that is capable of executing in just a few seconds. For that reason, it is important that web applications introduce some sort of limitation on the number of times a user can attempt to login before being locked out. For instance, iOS phones limit users to six attempts to enter the correct PIN before locking the user out for a certain amount of time. Otherwise, hackers could write a script to try every possible combination of numbers until the phone is inevitably unlocked; this is called a brute force attack. Developers must ensure that logic is in place in a web application's authentication system to account for such vulnerabilities. There are even third-party services that developers can include in their applications to prevent against attacks of this nature. Such services include Single Sign-On (SSO) solutions such as Security Assertion Markup Language (SAML) that offer a number of security advantages over writing a custom login service for a particular web application.

V. SQL Injection

Similar to XSS, SQL injection refers to an injection attack wherein a hacker can execute malicious SQL statements directly on a web application's database server. Structured Query Language (SQL) is a high-level database programming language designed for managing data held in relational databases (very often responsible for storing and providing content to websites and web applications) that is ubiquitous in the database community. Since so many websites are dependent on databases that can be queried and manipulated using the SQL programming language, this makes SQL injection attacks particularly detrimental to websites. Hackers can delete an entire database with just a single query, capable of eliminating years' worth of data that many businesses are dependent on. The good news about SQL injection is it is completely preventable if the correct steps are taken by the software development team. Similar to XSS, to prevent an injection attack, it is vital that the SQL compiler understands which portion of the input string needs to be treated as code, and which part of the string needs to be treated as variables. That is to protect against SQL injection, the developer must ensure that all user input data is treated as text and not as part of the command itself, as the hacker may try to insert text that includes SQL keywords to "inject" their own code. Fortunately, all server-side programming languages allow developers to create *parameterized queries*. A parameterized query is a means of pre-compiling a SQL statement (containing only the software developer's original code) and *then* passing in all user input in the form of parameters later on, thus ensuring that all user input is treated strictly as data values and not as SQL code. This completely prevents the possibility for hackers to directly manipulate databases via SQL

injection attacks. The main takeaway for developers: Ensure all SQL queries performed by a web application are parameterized queries.

Conclusion:

In conclusion, with the vast amount of personal information that is passed via web applications each day, it is vital that software developers and IT professionals ensure measures are in place to prevent against hacker attacks to ensure the safety of business and sensitive customer information. As evidence by recent political events, the role that hacking plays in today's society will only continue to increase. It is important that businesses and organizations identify potential security vulnerabilities with their web applications and continue to implement measures to counteract them. A secure web user experience begins with a well-designed and secure web application. It is the responsibility of software developers and technical leadership to continue to educate themselves on new hacker prevention techniques and work with management to plan and implement such measures.

BlackBerry is doing an excellent job of finding and eliminating potential seurity vulnerabilities. As an industry leader in the field of web and mobile security, it is my recommendation that BlackBerry continues to investigate new ways to eliminate potential security threats while adhering to industry best practices.

REFERENCES

"Brute Force Attack." *Techopedia*. Last Modified January 2017. https://www.techopedia.com/definition/18091/brute-force-attack

"Cross-site Scripting (XSS)." *Owasp.org*. Last Modified April 2016. https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

"Hillary Clinton Blames Russian Hackers, James Comey for Election Loss." V*ariety.com*. Last modified May 2017. http://variety.com/2017/biz/news/hilary-clinton-blames-russian-hackers-james-comey-election-loss-1202407033/

"How Will 2017 Security Breaches Stack Up?" *Revision/Legal*. Last modified February 2016. https://revisionlegal.com/data-breach/2017-security-breaches/

"HTTP - HyperText Transfer Protocol." *Webopedia.com*. Last Modified January 2017. http://www.webopedia.com/TERM/H/HTTP.html

"Information Leakage." *The Web Application Security Consortium*. Last Modified May 2010. http://projects.webappsec.org/w/page/13246936/Information%20Leakage

"Insufficient Transport Layer Protection." *Veracode*. Last Modified January 2017. https://www.veracode.com/security/insufficient-transport-layer-protection

"Preventing Cross-Site Scripting." *Youtube*. Last Modified August 2015. https://www.youtube.com/watch?v=0Oz645g3Wh0

"SQL Injection (SQLi)." *Acunetix*. Last Modified January 2017. https://www.acunetix.com/websitesecurity/sql-injection/

"The World's Most Trusted Mobile Security." *BlackBerry.com*. Last modified January 2017. https://ca.blackberry.com/enterprise/security

"Top 10 2010-A9-Insufficient Transport Layer Protection." *Owasp.org*. Last Modified October 2010. https://www.owasp.org/index.php/Top_10_2010-A9-Insufficient_Transport_Layer_Protection

"Web Applications Security Statistics Report." *WhiteHat Security*. Last Modified December 2016. https://info.whitehatsec.com/rs/675-YBI-674/images/WH-2016-Stats-Report-FINAL.pdf

"What is HTTPS?" *Instant SSL*. Last Modified January 2017. https://www.instantssl.com/ssl-certificate-products/https.html

"What is parameterized Query?" *Stack Overflow*. Last Modified January 2011. http://stackoverflow.com/questions/4712037/what-is-parameterized-query